Modern Engineering

January 26 | Day 13

Prudential Financial

GENERAL ASSEMBLY

LESSON ROADMAP

Containerization

Docker and Docker Images

Creating a Docker Image

Docker + Express

Docker + React + Express

MEF MODULE 5 DAY 13: Containerization with Docker

Schedule	
9:00–9:15 am	Welcome and Warm-Up
9:15–9:45 am	**Introduction to Docker, Docker Hub**
9:45 am–10:30 am	Dockerizing an Application
10:30–12:30 pm	**Dockerizing Express To Do App**
12:30–1:30 pm	Lunch
1:30-4:50 pm	**Dockerize Solo Application**
4:50–5:00 pm	Bring It Home

LEARNING OBJECTIVES

1. Explain the benefits of containerization in the context of a modern microservices application
2. Write a Dockerfile to create a Docker image
3. Containerize a react application
4. Apply Docker to a full-stack react application

Docker

Introduction to Docker

What is Docker?

Docker is a platform for building, deploying, and running applications in **containers**.

Docker creates a **container** from an **image,** which is essentially a *snapshot* of a software environment at a specific point in time.

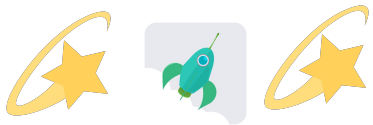
A **Dockerfile** contains instructions to create an image.

Docker Engine

The Docker Engine integrates with the OS to run the Docker containers; sits directly on top of the OS. It's very fast and inexpensive.

- **Docker Daemon:** The Docker daemon (**dockerd**) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. It does the heavy lifting and is a part of the Docker engine.
- **Docker Client (CLI):** The CLI uses Docker APIs to control or interact with the Docker daemon through scripting or direct CLI commands.

```
Dockerfile --> Image --> Container --> Docker Engine -->
```



Discussion:

The time before Docker...

10 Minutes

Before the widespread adoption of Docker, what did we do?

How did we deploy/manage our applications and services on the cloud?

Come off mute and discuss!

Some thoughts on “Before Docker”

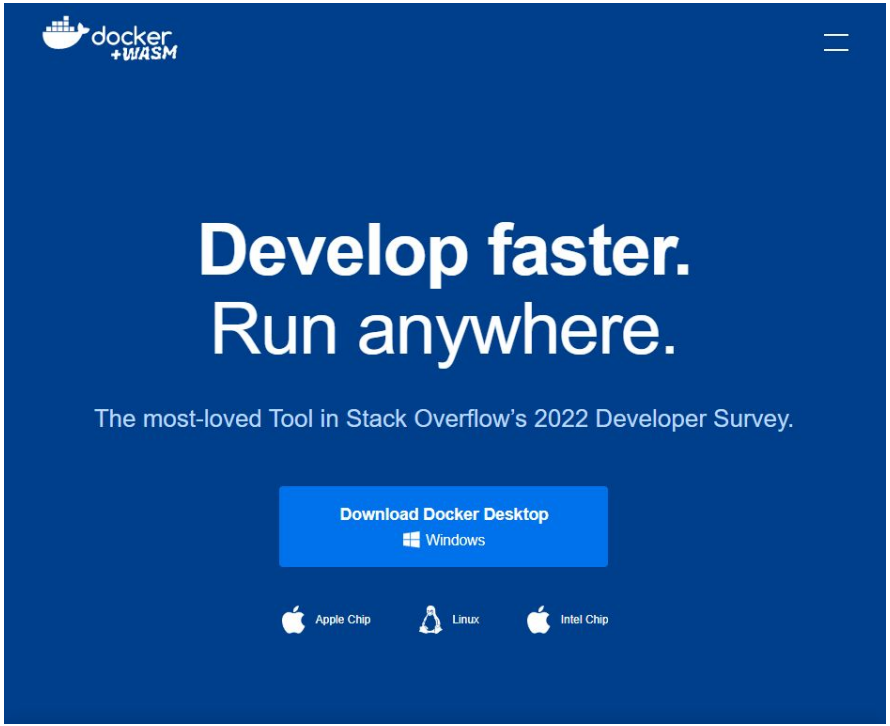
We relied on a combination of the following to manage our applications/services (which we still rely on, just not for everything!)

- Use of **Virtual Machines** (VMs)
- **Package managers** such as apt and yum, to manage installation of software on servers
- **Configuration management tools** (Ansible, Chef, Puppet etc.), to automate deployment and setup
- **Manual deployments**, with deployment scripts copied over to production servers

Docker

Docker Hub

Docker Hub



Docker Hub is a repository of free public images.

According to Docker, it is the world's largest library and community for container images.

You can access and create an account on Docker Hub here.

Source: Docker.com

Guided Walk-Through:
Docker Hub

30 minutes

1. Run: `docker run hello-world` in your terminal.
2. You should see:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

1. Scroll up to see `Unable to find image 'hello-world:latest' locally`
2. Now run the command again. You won't see the above line now. **Why not?**

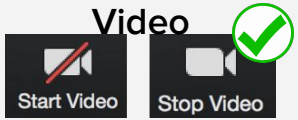
5:00

See You in 5 Minutes!

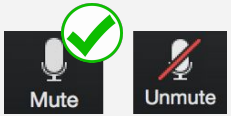
Have a question? Type in the chat.

While you're away:

Turn Off Video



Mute Your Microphone



Docker

Container Lifecycle

Container Lifecycle

Let’s look at life of a container.

To start up a new container from an image we use the **docker run** command. docker run creates a new container and starts it. But creating a container and starting it up are 2 different processes.

docker run = docker create + docker start

When a container is created, the file system is prepped for use in the new container. To start the container means executing the startup command that comes with the image.

Let’s see it in action.

Guided Walk-Through:
Container Lifecycle

10 minutes

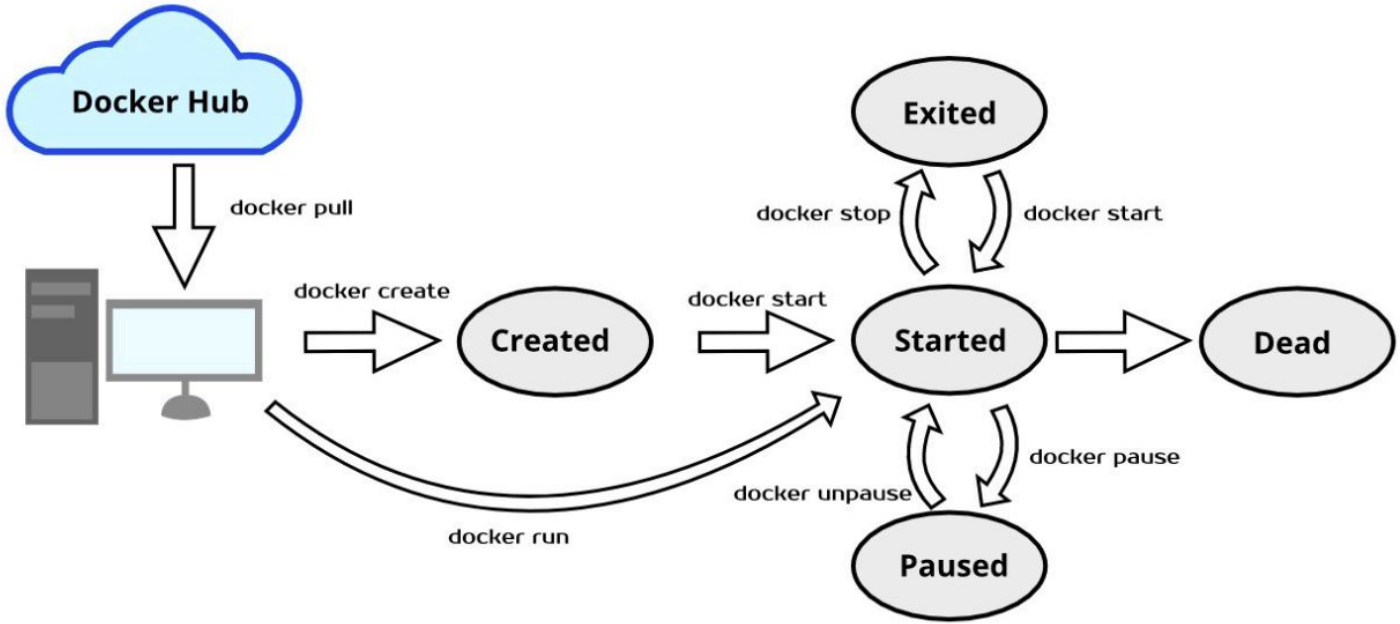
1. Run: `sudo docker create hello-world` in your terminal.
2. It should output a container ID e.g. `2ce9bdb14238e8a6ba2ac68964ee8ca48e93e9cbf`
3. Run docker start with your container ID, e.g.

   ```
   sudo docker start -a 2ce9bdb14238e8a6ba2ac68964ee8ca48e93e9cbf
   ```

4. Nice! Let visualise what just happened.

NOTE: If you get an error in your Terminal it could be because Docker is not running in your VM, try this: `sudo service docker start`

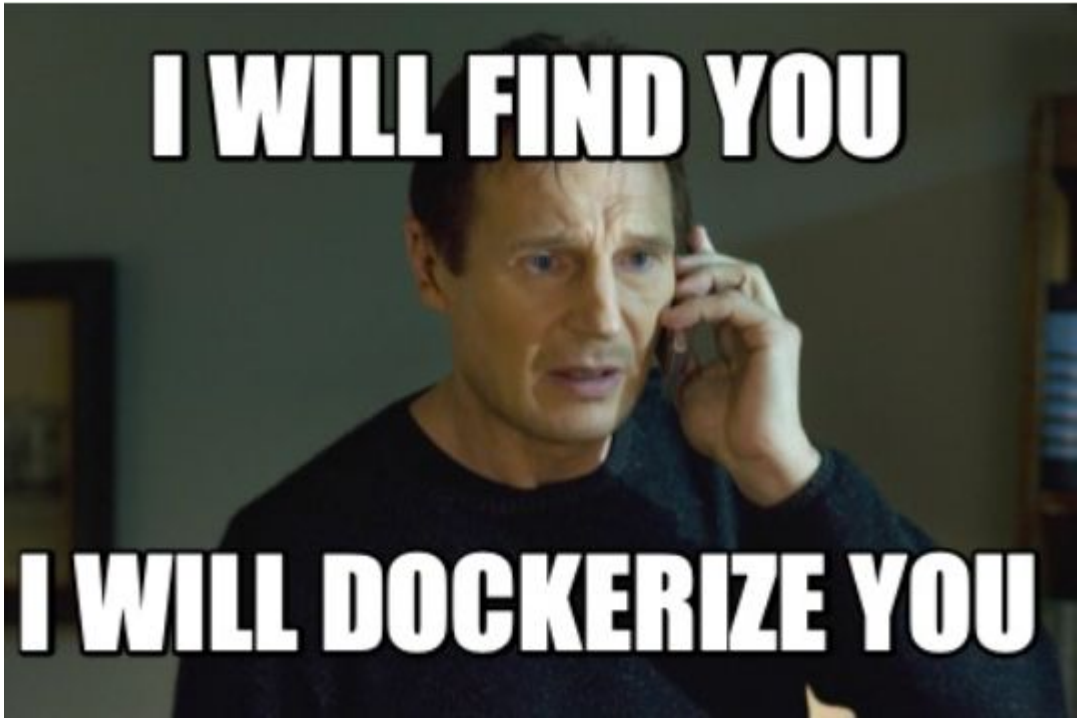
Container Lifecycle Visualized



Source: digitaljournal.com

Docker

Dockerizing an App



So far, we have used an existing image to run a container.

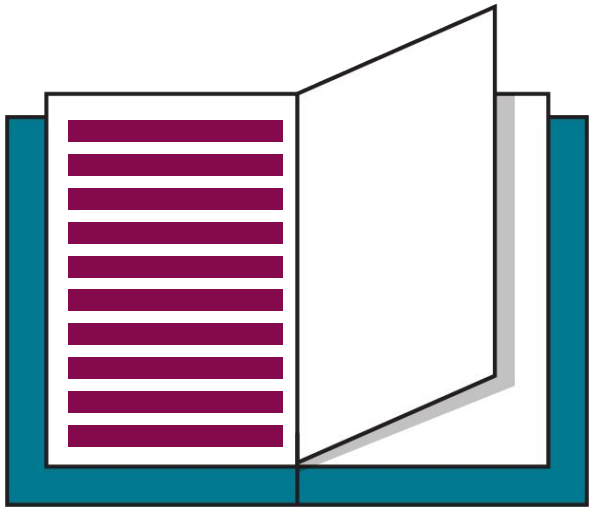
But we haven't talked about how to create an image (or “dockerize”) our app.

Source: memecreator.org

Dockerizing our app

According to docs, a Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

It is used by docker to build images automatically by reading instructions from it.



Dockerizing our app

A few things to note...

- You can specify details like, the operating system, languages, etc.
- Explain what the container will do when it's run. You can create your own or use premade ones.
- Each instruction adds a new layer to the container’s filesystem
- You can include steps such as files into the container

Dockerizing our app

Here's an example Dockerfile for a Java Web App:

Dockerfile

```
FROM tomcat:8.0-alpine

LABEL maintainer="tristan.hall@generalassemb.ly"

ADD JavaWebApp.war /usr/local/tomcat/webapps/

EXPOSE 8080

CMD ["catalina.sh", "run"]
```


Docker and Kubernetes

It's also important to say what Docker **isn't.** It's a platform for building and running containers, but doesn't automate the deployment, scale, and management of containers.

For that, we use an **orchestration** layer that manages Docker across multiple machines. Examples include Kubernetes, Nomad, Mesos, Swarm, and others

Guided Walk-Through:

Dockerizing Express To Do API

90 minutes

Here we will create a Dockerfile for our Express application.

You will use the Express you built in the previous track for this lab.

Detailed instructions can be found in GitHub.

Break Time

Solo Exercise:

Dockerize Your Express + React App

120 minutes

Now, on your own, you will create 3 Dockerfiles for your Express + React application and database.

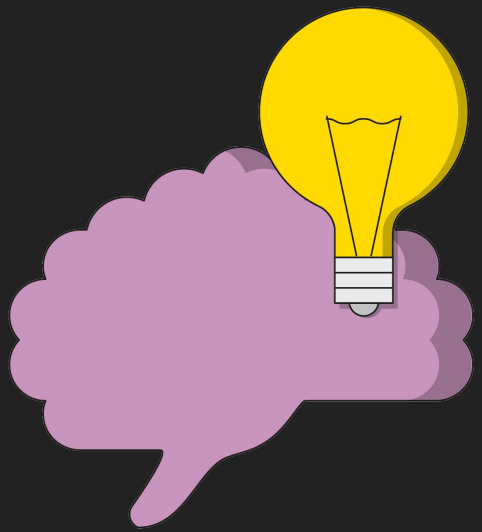
You can use your independent Full-Stack react app, or any other application using both Express and React in tandem with a database.

You will need 3 Dockerfiles: one for the *backend, db,* and *frontend* folders.

You can use today's dockerize todo app lesson for reference.

Here is a README with lab details.

Knowledge Check!



A. A **Docker container** is created from a/an _____.
B. To start up a **new container from an image** we use the _____ command.

Solo Exercise:

Daily Exit Ticket

3 minutes

- Please take a moment to give us your feedback after today's course!
- Use the QR code or follow the link: https://bit.ly/pruMEFc2

bitly

Scan Me!


GA